Processamento de dados usando *dplyr*

Vanderlei Júlio Debastiani (vanderleidebastiani@yahoo.com.br)

20 Maio 2020

Índice de conteúdo

1 Introdução

O processamento de dados é uma tarefa importante a ser realizada na ciência. Após os dados serem coletados, é preciso tratar esses dados de maneira sistemática para extrair informações úteis. O processamento de dados é um processo dinâmico, com o objetivo de ordenar, classificar ou efeturar transformações para obter resultados que possam ser mais facilmente interpretados ou usados em outras análises estatísticas.

Essa organização e reorganização dos conjuntos de dados normalmente é realizada em programas do tipo editor de planilhas, entretanto o processamento e manipulação de dados também pode ser realizadas no ambiente **R**. Neste, o processamento pode ser realizado de uma maneira eficiente, aproveitando das demais ferramentas estatísticas e gráficas do ambiente.

O pacote *dplyr* é um pacote de processamento e manipulação de dados. O pacote implementa funções que são, em certo modo, redundante com as funções básicas (pacote *base*) do **R**, mas oferece um conjunto de funções desenvolvidas para ajudar a resolver os problemas mais comuns enfrentados no processamento de dados. Ainda, o pacote utiliza uma forma gramatical da manipulação dos dados, devido a sua forma mais gramatical seu uso é otimizado no *RStudio* em função dos recursos de autocomplemento.

Este texto aborda os aspectos básicos do pacote *dplyr*. O pacote está inserido em um conjunto maiores de pacotes chamado *tidyverse*, pacotes que seguem uma lógica similar, como por exemplo *magrittr*, *ggplot2* e *tibble*. O tutorial requer conhecimento básico sobre **R**, neste os código são apenas brevemente comentados sendo que a ideia é que o leitor seja capaz de reproduzir os comandos aqui contidos, e busque mais informações sobre os argumentos extras nas páginas de ajuda de cada função.

O texto está dividido em três partes: apresentação dos dados utilizados como exemplo, os operadores de encadeamento e processamento de dados. Ainda, no final do texto, é apresentado um guia de ajuda rápida mostrando as principais funções e operadores apresentados aqui.

2 Tabelas de dados

O *dplyr* é focado no processamento de dados em *data.frames*. Os *data.frames* são tabelas que armazenam um ou mais vetores de dados, que como características elas possuem duas dimensões, linhas e colunas. Cada coluna, será um vetor, podendo portando armazenar vetores de diferentes tipos (*numeric*, *character*, *logical* ou *factor*). Pressupoe-se que cada variável esteja na sua própria coluna, que cada observação/registro esteja na sua própria linha e que todas as células estejam preenchidas. O precessamento de dados usando *dplyr* preserva automaticamente a estrutura geral do *data.frame* conforme as linhas ou colunas são manipuladas.

2.1 Dados de exemplo

Nesse tutorial os dados de exemplo são provenientes de Wohlschag (1957) sendo estes medidas metabólicas feitas em indivíduos de peixes das espécies *Coregonus sardinella* e *C. autumnalis* em dois ambientes: Elson Lagoon (marinho) e Ikroavik Lake (água doce).

Variávels inclúidas (colunas):
- Especie (C.sardinella ou C.autumnalis);
- Ambiente (Marinho ou A.doce);
- ConsumoO2 (Consumo de O2 em mg);
- Massa (g);
- ConsumoO2Kg (mg O2 consumido/hora/kg);
- Rotacao (R.P.M.);
- Temperatura (Celsius).

Os dados podem ser baixados em github.com/vanderleidebastiani/tutoriais.

```
# Carregar dados
# dados <- read.csv("Metabolismo_Coregonus.csv", header = TRUE)
urlRemote  <- "https://raw.githubusercontent.com/"
pathGithub <- "vanderleidebastiani/tutoriais/master/Dados/"
fileName   <- "Metabolismo_Coregonus.csv"
dados <- read.csv(paste0(urlRemote, pathGithub, fileName), header = TRUE)
```

```
dados
        Especie Ambiente Consumo02 Massa Consumo02kg Rotacao Temperatura
1  C.sardinella  Marinho    38.868   320       2.084     0.0         6.7
2  C.sardinella  Marinho    24.882   109       2.358    12.5        10.8
3  C.sardinella  Marinho    29.292   143       2.311     4.5        10.3
4  C.sardinella  Marinho    45.618   225       2.307     6.0        10.3
5  C.sardinella  Marinho    35.694   174       2.312     6.0         8.9
6  C.sardinella  Marinho    38.976   188       2.317     3.5         8.9
7  C.sardinella  Marinho    33.498   143       2.370     0.0         9.4
8  C.sardinella  Marinho    31.524   104       2.482     5.0         9.4
9  C.sardinella   A.doce    20.196   138       2.165     1.0         9.2
10 C.sardinella   A.doce    39.690   365       2.036     4.0         9.2
11 C.sardinella   A.doce    25.812   140       2.266     8.0        10.8
12 C.sardinella   A.doce    37.404   183       2.310    10.0        10.6
13 C.sardinella   A.doce    33.396   221       2.179     5.0         9.4
14 C.sardinella   A.doce    29.100   204       2.154     7.0         9.4
15 C.sardinella   A.doce    36.654   256       2.156     5.5         9.7
16 C.sardinella   A.doce    11.016   135       1.912     2.0         9.7
17 C.sardinella   A.doce    20.340   180       2.053     5.0         9.4
18 C.sardinella   A.doce    41.616   362       2.061     5.0         8.3
19 C.sardinella   A.doce    27.858   231       2.081     3.0         8.9
20 C.sardinella   A.doce    30.546   216       2.151     2.5         8.9
21 C.sardinella   A.doce    16.758   129       2.114     7.5         9.2
22 C.sardinella   A.doce    25.320   129       2.293     7.0         9.2
23 C.autumnalis  Marinho    36.852   157       2.371     0.0         7.2
24 C.autumnalis  Marinho    13.494   124       2.037     0.0         7.2
25 C.autumnalis  Marinho    24.648   130       2.278     0.0         6.7
26 C.autumnalis  Marinho    32.274   234       2.140     5.0         7.2
```

3 Encadeamento

3.1 %>% - Encadeamento (*pipe*)

A estutura padrão dos códigos **R** tende a produzir funções aninhadas. O operador de encadeamento, no inglês *pipe*, implementado no pacote *magrittr*, altera a estrutura padrão das funções. O operador %>% encaminhara um objeto (valor ou resultado) para a próxima função. No *RStudio* o operador pode ser acessado pelo atalho pelo **CTR + SHIFT + M** no Windows e pelo atalho **CMD + SHIFT + M** no macOS.

Figura 1: Fluxograma pipe.

Por exemplo, para calcular o logaritmo de um valor:

```
require(magrittr)

# Linguagem padrão
log(x = 12, base = 2)
[1] 3.584963
```

```
# Usando pipe
12 %>% log(base = 2)
[1] 3.584963
```

Por padrão usando o operador *pipe* o objeto passado pelo operador sempre será o primeiro argumento. Para alterar esse comportamento é preciso sinalizar com **"."** o argumento que o operador *pipe* está passando.

```
# Objeto passado para x
12 %>% log(x = ., base = 2)
[1] 3.584963

# Objeto passado para base
2 %>% log(x = 12, base = .)
[1] 3.584963

# Note que o primeiro argumento da função sempre será passado pela função pipe ...
12 %>% log(base = 2)
[1] 3.584963

# ... então o segundo argumento da função pode ser passado sem ser especificado
12 %>% log(2)
[1] 3.584963
```

A principal vantagem do operador *pipe* é gerar essa sequência de processos encadeados, onde a saída de uma função é utilizada como entrada de uma função seguinte. O operador tem como utilizadade evitar o uso de funções aninhadas, minimizar a necessidade de variáveis locais (temporárias) e permitir a adição (ou supressão) de etapas em qualquer parte do código. O operador *pipe* pode ser usado com qualquer função, mas tem uso preferencial no processamento de dados usando o pacote *dplyr*, uma vez que normalmente são realizadas várias tarefas em sequência, como selecionar variáveis, filtrar linhas ou aplicar transformações nas variáveis. Por outro lado, como um ponto negativo, o operador não funciona muito bem quando vários objetos estão envolvidos, tanto de entradas quanto de saídas de resultados.

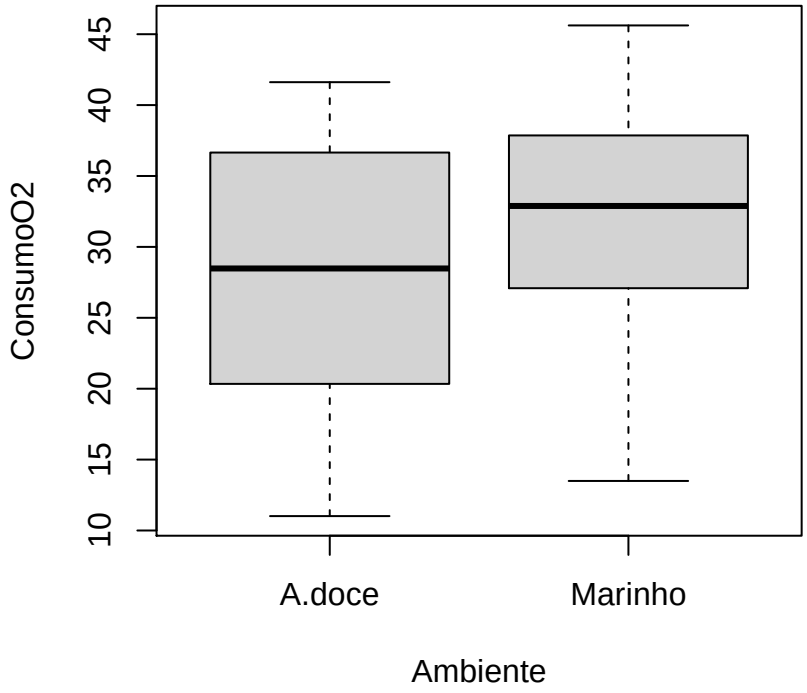
```
# Para usar as opções padrão de qualquer função não é preciso usar os parênteses ...
dados %>% head
       Especie Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
1 C.sardinella  Marinho    38.868   320       2.084     0.0         6.7
2 C.sardinella  Marinho    24.882   109       2.358    12.5        10.8
3 C.sardinella  Marinho    29.292   143       2.311     4.5        10.3
4 C.sardinella  Marinho    45.618   225       2.307     6.0        10.3
5 C.sardinella  Marinho    35.694   174       2.312     6.0         8.9
6 C.sardinella  Marinho    38.976   188       2.317     3.5         8.9

# ... mas não há problemas em usar
dados %>% head()
       Especie Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
1 C.sardinella  Marinho    38.868   320       2.084     0.0         6.7
2 C.sardinella  Marinho    24.882   109       2.358    12.5        10.8
3 C.sardinella  Marinho    29.292   143       2.311     4.5        10.3
4 C.sardinella  Marinho    45.618   225       2.307     6.0        10.3
5 C.sardinella  Marinho    35.694   174       2.312     6.0         8.9
6 C.sardinella  Marinho    38.976   188       2.317     3.5         8.9

# O segundo argumento da função original pode ser passado sem especificar o argumento
dados %>% head(2)
```

```
       Especie Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
1 C.sardinella  Marinho    38.868   320       2.084     0.0         6.7
2 C.sardinella  Marinho    24.882   109       2.358    12.5        10.8
```

```
# O . pode ser usado para passar o argumento para outro argumento que não o primeiro
dados %>% boxplot(ConsumoO2~Ambiente, data = .)
```




```
# Para indexar é preciso usa o .
dados %>% .$Ambiente
 [1] "Marinho" "Marinho" "Marinho" "Marinho" "Marinho" "Marinho" "Marinho" "Marinho"
 [9] "A.doce"  "A.doce"  "A.doce"  "A.doce"  "A.doce"  "A.doce"  "A.doce"  "A.doce"
[17] "A.doce"  "A.doce"  "A.doce"  "A.doce"  "A.doce"  "A.doce"  "Marinho" "Marinho"
[25] "Marinho" "Marinho"
dados %>% .[1,]
       Especie Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
1 C.sardinella  Marinho    38.868   320       2.084       0         6.7
dados %>% .[,1]
 [1] "C.sardinella" "C.sardinella" "C.sardinella" "C.sardinella" "C.sardinella"
 [6] "C.sardinella" "C.sardinella" "C.sardinella" "C.sardinella" "C.sardinella"
[11] "C.sardinella" "C.sardinella" "C.sardinella" "C.sardinella" "C.sardinella"
[16] "C.sardinella" "C.sardinella" "C.sardinella" "C.sardinella" "C.sardinella"
[21] "C.sardinella" "C.sardinella" "C.autumnalis" "C.autumnalis" "C.autumnalis"
[26] "C.autumnalis"
dados %>% .[2,1:4]
       Especie Ambiente ConsumoO2 Massa
2 C.sardinella  Marinho    24.882   109

# Aplicar múltiplos encadeamentos
# Selecionar variável Massa, calcular o log do vetor e então calcular a média
dados %>%
    .$Massa %>%
```

```
    log %>%
    mean
[1] 5.18491
```

3.2 Outras funções do pacote *magrittr*

3.2.1 %$% - Selecionar a variável de interesse

Muitas funções como por exemplo *boxplot()* e *lm()* permitem selecionar as variáveis pelos nomes das colunas usando o argumento *data*, entretando o argumento data não está disponível em todas as funções. Usando o operador **%$%** é possível selecionar as variáveis pelo nome sem usar a função *attach()*.

```
# Selecionar uma variável ...
dados %$% sum(Temperatura)
[1] 234.9

# ... ou múltiplas variávels
dados %$% cor(Consumo02, Temperatura)
[1] -0.01111897
```

3.2.2 %<>% - Atribur diretamente

O operador %<>% permite atribuir diretamente o resultado de uma função no objeto passado para o operador.

```
# Note a classe da variável Ambiente...
dados %>% str
'data.frame':   26 obs. of  7 variables:
 $ Especie    : chr  "C.sardinella" "C.sardinella" "C.sardinella" "C.sardinella" ...
 $ Ambiente   : chr  "Marinho" "Marinho" "Marinho" "Marinho" ...
 $ Consumo02  : num  38.9 24.9 29.3 45.6 35.7 ...
 $ Massa      : int  320 109 143 225 174 188 143 104 138 365 ...
 $ Consumo02kg: num  2.08 2.36 2.31 2.31 2.31 ...
 $ Rotacao    : num  0 12.5 4.5 6 6 3.5 0 5 1 4 ...
 $ Temperatura: num  6.7 10.8 10.3 10.3 8.9 8.9 9.4 9.4 9.2 9.2 ...

# ... o operador já atribui a alteração da classe da variável...
dados$Ambiente %<>% as.factor

# ... Nova classe da variável Ambiente
dados %>% str
'data.frame':   26 obs. of  7 variables:
 $ Especie    : chr  "C.sardinella" "C.sardinella" "C.sardinella" "C.sardinella" ...
 $ Ambiente   : Factor w/ 2 levels "A.doce","Marinho": 2 2 2 2 2 2 2 2 1 1 ...
 $ Consumo02  : num  38.9 24.9 29.3 45.6 35.7 ...
 $ Massa      : int  320 109 143 225 174 188 143 104 138 365 ...
 $ Consumo02kg: num  2.08 2.36 2.31 2.31 2.31 ...
 $ Rotacao    : num  0 12.5 4.5 6 6 3.5 0 5 1 4 ...
 $ Temperatura: num  6.7 10.8 10.3 10.3 8.9 8.9 9.4 9.4 9.2 9.2 ...
```

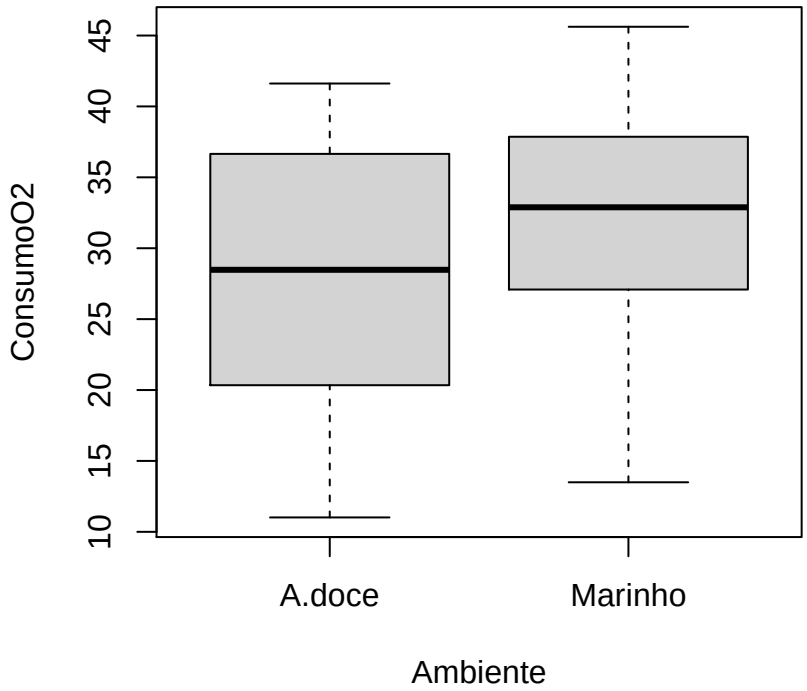
3.2.3 %T>% - Abrir braço no fluxo de encadeamento

O operador **%T>%** permite criar uma espécie de braço da sequência de encadeamento para canalizar os resultados para uma função mantendo o resultado original. Esse operador é útil quando aplicado a uma

função para visualizar os resultados antes de terminar a sequência *pipe*, como por exemplo nas funções *plot()* ou *print()*.

```
# A função summary() é aplicada ao objeto resultante da função boxplot...
dados %>%
    boxplot(ConsumoO2~Ambiente, data = .) %>%
    summary()
      Length Class  Mode
stats 10     -none- numeric
n      2     -none- numeric
conf   4     -none- numeric
out    0     -none- numeric
group  0     -none- numeric
names  2     -none- character

# ... a função summary() é aplicada no objeto dados
dados %T>%
    boxplot(ConsumoO2~Ambiente, data = .)  %>%
    summary()
```




```
  Especie             Ambiente    ConsumoO2         Massa        ConsumoO2kg
Length:26          A.doce :14   Min.   :11.02   Min.   :104.0   Min.   :1.912
Class :character   Marinho:12   1st Qu.:24.99   1st Qu.:135.8   1st Qu.:2.091
Mode  :character                Median :31.04   Median :177.0   Median :2.172
                                Mean   :30.05   Mean   :190.0   Mean   :2.204
                                3rd Qu.:36.80   3rd Qu.:224.0   3rd Qu.:2.311
                                Max.   :45.62   Max.   :365.0   Max.   :2.482

    Rotacao        Temperatura
Min.   : 0.000   Min.   : 6.700
1st Qu.: 2.125   1st Qu.: 8.900
Median : 5.000   Median : 9.200
Mean   : 4.423   Mean   : 9.035
```

```
 3rd Qu.: 6.000   3rd Qu.: 9.625
 Max.   :12.500   Max.   :10.800
```

4 Processamento de dados usando *dplyr*

No processamento básico de dados os principais objetivos são filtrar ou selecionar conjuntos de registros e variáveis, gerar novas variáveis a partir de variáveis existentes, bem como ordenar e aplicar estatísticas descritivas. O pacote *dplyr* conta com várias funções, com descate as apresentadas abaixo:

- **filter()** - Filtrar os observações/registros com base em seus valores;
- **select()** - Selecionar variáveis com base em seus nomes;
- **mutate()** - Adicionar novas variáveis que são funções de variáveis existentes;
- **arrange()** - Alterar a ordem das linhas;
- **group_by()** - Agrupar registros para aplicar funções por grupos;
- **summarise()** - Reduzir vários valores para um estátisticas descritivas.

No pacote *dplyr* a maioria das funções tem como primeiro argumento o *data.frame* (argumentos *.data* ou *.tbl*) com os dados. Essa padronização dos argumentos das funções facilita o uso do operador *pipe*. Nesse tutorial o objeto contendo os dados é sempre passado usando o operador *pipe*, mas as funções também poderiam ser executadas de maneira convencional.

4.1 Tabelas de dados (*tibble*)

Os *tibbles*, classe *tbl_df*, são reimplementação dos *data.frame* feitas pelo pacote *tibble*. Nesta classe, o processamento não altera os nomes ou tipos de variáveis, não fazem correspondência parcial das variáveis e reclamam quando uma variável não existe. Os *tibbles* também têm um método *print()* próprio, sendo fáceis de usar com grandes conjuntos de dados já que apenas as primeiras linhas e colunas são mostradas. O método *print()*, de mandeira complementar, também mostra um resumo da quantidade de linhas e colunas das tabelas de dados e o tipo de cada variável.

Várias funções do pacote *dplyr* retornam *data.frame*, mas algumas delas retornam apenas *tibble*, entretanto é possível converter-los usando as funções **as.tbl** e **as.data.frame**.

```
require(dplyr)

# Converter dada.frame em tbl_df
dados <- as.tbl(dados)

# O objeto pertence a múltiplas classes
class(dados)
[1] "tbl_df"     "tbl"        "data.frame"

# Método print. Note as informações adicionais apresentadas
dados
# A tibble: 26 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       38.9   320        2.08     0           6.7
 2 C.sardinella Marinho       24.9   109        2.36    12.5        10.8
 3 C.sardinella Marinho       29.3   143        2.31     4.5        10.3
 4 C.sardinella Marinho       45.6   225        2.31     6          10.3
 5 C.sardinella Marinho       35.7   174        2.31     6           8.9
 6 C.sardinella Marinho       39.0   188        2.32     3.5         8.9
 7 C.sardinella Marinho       33.5   143        2.37     0           9.4
 8 C.sardinella Marinho       31.5   104        2.48     5           9.4
```

```
 9 C.sardinella A.doce        20.2   138        2.16     1           9.2
10 C.sardinella A.doce        39.7   365        2.04     4           9.2
# ... with 16 more rows
```

4.2 Variações das funções e seleção de variáveis

A maioria das funções do pacote *dplyr* possui variáções gramaticais. Essas variações são usadas para aplicar as funções principais a conjuntos de variáveis da tabela de acordo com alguns critérios. As variações seguem o mesmo padrão entre todas as funções, tendo como sufixo:

- **_all** - Aplicam ou forçam a aplicação em todas as variáveis da tabela;
- **_if** - Aplicam as funções condicionada as variáveis (definidas pelo argumento *.predicate*). Geralmente usada em conjunto com as funções **is.numeric**, **as.factor**, ...;
- **_at** - Aplicam as funções a algumas variáveis selecionadas (definidas pelo argumento *.vars*).

```
# Variações
xxxxx_all(.data, ...)
xxxxx_if(.data, .predicate, ...)
xxxxx_at(.data, .vars, ...)
```

A seleção de variáveis da tabela também pode ser feita de maneira gramatical usando a função **vars()** ou funções auxiliares.

```
# Incluir ou excluir variáveis pelo mome ou posição
vars(...)

# Selecionar variáveis cujo nome começam por ...
starts_with(match, ...)

# ... terminam com ...
ends_with(match, ...)

# ... que contenham ...
contains(match, ...)

# ... que correspondem a uma expressão ...
matches(match, ...)

# ... que correspondam a um intervalo numérico
num_range(prefix, range, ...)
```

4.3 filter() - Filtrar linhas

A função **filter()** permite filtrar linhas, que contêm as observações/registros, com base em seus valores, já a função **slice()** permite selecionar observações com base na posição das linhas na tabela.

```
# Funções principais
slice(.data, ...)
filter(.data, ...)

# Variações
filter_all(.data, .vars_predicate, ...)
filter_if(.data, .predicate, .vars_predicate, ...)
filter_at(.data, .vars, .vars_predicate, ...)
```

```
# Principais funções associadas
==, >, >=, <, <=, !=, &, |, is.na()
between(x, left, right)
near(x, y, ...)
all_vars(expr)
any_vars(expr)
```

4.3.1 slice() - Filtrar linhas usando posição

```
# Fornecer as posições das linhas para filtrar...
dados %>% slice(2:4)
# A tibble: 3 x 7
  Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
  <chr>        <fct>         <dbl> <int>       <dbl>   <dbl>       <dbl>
1 C.sardinella Marinho        24.9   109        2.36    12.5        10.8
2 C.sardinella Marinho        29.3   143        2.31     4.5        10.3
3 C.sardinella Marinho        45.6   225        2.31     6          10.3

# ... não é preciso concatenar
dados %>% slice(2,4,9)
# A tibble: 3 x 7
  Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
  <chr>        <fct>         <dbl> <int>       <dbl>   <dbl>       <dbl>
1 C.sardinella Marinho        24.9   109        2.36    12.5        10.8
2 C.sardinella Marinho        45.6   225        2.31     6          10.3
3 C.sardinella A.doce         20.2   138        2.16     1           9.2
```

4.3.2 filter() - Filtrar linhas usando critérios lógicos

```
# Filtrar usando critério lógico aplicado a uma variável ...
dados %>% filter(Temperatura>10)
# A tibble: 5 x 7
  Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
  <chr>        <fct>         <dbl> <int>       <dbl>   <dbl>       <dbl>
1 C.sardinella Marinho        24.9   109        2.36    12.5        10.8
2 C.sardinella Marinho        29.3   143        2.31     4.5        10.3
3 C.sardinella Marinho        45.6   225        2.31     6          10.3
4 C.sardinella A.doce         25.8   140        2.27     8          10.8
5 C.sardinella A.doce         37.4   183        2.31    10          10.6

# ... ou aplicado à várias variáveis
dados %>% filter(Temperatura>10 & Ambiente == "Marinho")
# A tibble: 3 x 7
  Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
  <chr>        <fct>         <dbl> <int>       <dbl>   <dbl>       <dbl>
1 C.sardinella Marinho        24.9   109        2.36    12.5        10.8
2 C.sardinella Marinho        29.3   143        2.31     4.5        10.3
3 C.sardinella Marinho        45.6   225        2.31     6          10.3

# Filtrar usando função between
dados %>% filter(between(Temperatura, left = 9, right = 10))
```

```
# A tibble: 11 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       33.5   143        2.37     0           9.4
 2 C.sardinella Marinho       31.5   104        2.48     5           9.4
 3 C.sardinella A.doce        20.2   138        2.16     1           9.2
 4 C.sardinella A.doce        39.7   365        2.04     4           9.2
 5 C.sardinella A.doce        33.4   221        2.18     5           9.4
 6 C.sardinella A.doce        29.1   204        2.15     7           9.4
 7 C.sardinella A.doce        36.7   256        2.16     5.5         9.7
 8 C.sardinella A.doce        11.0   135        1.91     2           9.7
 9 C.sardinella A.doce        20.3   180        2.05     5           9.4
10 C.sardinella A.doce        16.8   129        2.11     7.5         9.2
11 C.sardinella A.doce        25.3   129        2.29     7           9.2

# Filtro condicionar. Aplicado apenas em variáveis numéricas
# A função all_vars retorna registros apenas se todos são verdadeiros
dados %>% filter_if(is.numeric, all_vars(. > 2.3))
# A tibble: 7 x 7
  Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
  <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
1 C.sardinella Marinho       24.9   109        2.36    12.5        10.8
2 C.sardinella Marinho       29.3   143        2.31     4.5        10.3
3 C.sardinella Marinho       45.6   225        2.31     6          10.3
4 C.sardinella Marinho       35.7   174        2.31     6           8.9
5 C.sardinella Marinho       39.0   188        2.32     3.5         8.9
6 C.sardinella Marinho       31.5   104        2.48     5           9.4
7 C.sardinella A.doce        37.4   183        2.31    10          10.6

# Fitro selecitivo. Aplicado apenas nas variáveis selecionadoas pela função var
# A função any_vars retorna registro se algum for verdadeiro
dados %>% filter_at(vars(Temperatura, Rotacao), any_vars(. == 0))
# A tibble: 5 x 7
  Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
  <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
1 C.sardinella Marinho       38.9   320        2.08       0         6.7
2 C.sardinella Marinho       33.5   143        2.37       0         9.4
3 C.autumnalis Marinho       36.9   157        2.37       0         7.2
4 C.autumnalis Marinho       13.5   124        2.04       0         7.2
5 C.autumnalis Marinho       24.6   130        2.28       0         6.7
```

4.3.3 Filtrar (extrar) aleatoriamente

As funções **sample_n()** e **sample_frac()** permitem filtrar/amostrar linhas aleatoriamente, enquanto que a função **top_n()** permite extrair os registros de acordo com os maiores valores de uma determinada variável.

```
# Funções principais
sample_n(.data, size, replace = FALSE, ...)
sample_frac(.data, size, replace = FALSE, ...)
top_n(.data, n, wt)
```

```
# Extrair n linhas aleatoriamete
dados %>% sample_n(size = 5, replace = FALSE)
# A tibble: 5 x 7
  Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
  <chr>        <fct>         <dbl> <int>       <dbl>   <dbl>       <dbl>
1 C.autumnalis Marinho        32.3   234        2.14     5           7.2
2 C.sardinella A.doce         25.8   140        2.27     8          10.8
3 C.sardinella A.doce         36.7   256        2.16     5.5         9.7
4 C.sardinella A.doce         20.3   180        2.05     5           9.4
5 C.sardinella Marinho        24.9   109        2.36    12.5        10.8

# Extrair 10% das linhas aleatoriamente
dados %>% sample_frac(size = 0.1, replace = FALSE)
# A tibble: 3 x 7
  Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
  <chr>        <fct>         <dbl> <int>       <dbl>   <dbl>       <dbl>
1 C.sardinella A.doce         20.3   180        2.05       5         9.4
2 C.sardinella Marinho        38.9   320        2.08       0         6.7
3 C.sardinella A.doce         25.3   129        2.29       7         9.2

# Extrair n linhas do maiores valores de determinada variável
dados %>% top_n(n = 5, wt = ConsumoO2kg)
# A tibble: 5 x 7
  Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
  <chr>        <fct>         <dbl> <int>       <dbl>   <dbl>       <dbl>
1 C.sardinella Marinho        24.9   109        2.36    12.5        10.8
2 C.sardinella Marinho        39.0   188        2.32     3.5         8.9
3 C.sardinella Marinho        33.5   143        2.37     0           9.4
4 C.sardinella Marinho        31.5   104        2.48     5           9.4
5 C.autumnalis Marinho        36.9   157        2.37     0           7.2
```

4.4 select() e rename() - Selecionar e renomear colunas

A função **select()** permite selecionar colunas/variáveis das tabelas com base nos nomes ou posição das colunas. A função **pull()** permite a seleção de uma variável já transformando em um vetor e a função **rename()** renomerar as colunas.

```
# Funções principais
pull(.data, var)
select(.data, ...)
rename(.data, ...)

# Variações
select_all(.data, .funs = list(), ...)
select_if(.data, .predicate, .funs = list(), ...)
select_at(.data, .vars, .funs = list(), ...)
rename_all(.data, .funs = list(), ...)
rename_if(.data, .predicate, .funs = list(), ...)
rename_at(.data, .vars, .funs = list(), ...)

# Principais funções associadas
vars(...)
starts_with(match, ...)
ends_with(match, ...)
```

```
contains(match, ...)
matches(match, ...)
num_range(prefix, range, ...)
```

4.4.1 pull() - Selecionar coluna e retornar um vetor

```
# Selecionar variável e retornar um vetor
dados %>% pull(Consumo02)
 [1] 38.868 24.882 29.292 45.618 35.694 38.976 33.498 31.524 20.196 39.690 25.812 37.404
[13] 33.396 29.100 36.654 11.016 20.340 41.616 27.858 30.546 16.758 25.320 36.852 13.494
[25] 24.648 32.274
```

4.4.2 select() - Selecionar coluna

```
# Selecionar variáveis usando posição...
dados %>% select(1:3)
# A tibble: 26 x 3
   Especie      Ambiente Consumo02
   <chr>        <fct>        <dbl>
 1 C.sardinella Marinho       38.9
 2 C.sardinella Marinho       24.9
 3 C.sardinella Marinho       29.3
 4 C.sardinella Marinho       45.6
 5 C.sardinella Marinho       35.7
 6 C.sardinella Marinho       39.0
 7 C.sardinella Marinho       33.5
 8 C.sardinella Marinho       31.5
 9 C.sardinella A.doce        20.2
10 C.sardinella A.doce        39.7
# ... with 16 more rows

# ... não é nescessário concatenar
dados %>% select(1, 3)
# A tibble: 26 x 2
   Especie      Consumo02
   <chr>            <dbl>
 1 C.sardinella      38.9
 2 C.sardinella      24.9
 3 C.sardinella      29.3
 4 C.sardinella      45.6
 5 C.sardinella      35.7
 6 C.sardinella      39.0
 7 C.sardinella      33.5
 8 C.sardinella      31.5
 9 C.sardinella      20.2
10 C.sardinella      39.7
# ... with 16 more rows

# Selecionar usando nomes...
dados %>% select(Ambiente, Massa)
# A tibble: 26 x 2
   Ambiente Massa
```

```
   <fct>    <int>
 1 Marinho    320
 2 Marinho    109
 3 Marinho    143
 4 Marinho    225
 5 Marinho    174
 6 Marinho    188
 7 Marinho    143
 8 Marinho    104
 9 A.doce     138
10 A.doce     365
# ... with 16 more rows

# ... ou sequências em nomes
dados %>% select(Ambiente:Massa)
# A tibble: 26 x 3
   Ambiente ConsumoO2 Massa
   <fct>        <dbl> <int>
 1 Marinho       38.9   320
 2 Marinho       24.9   109
 3 Marinho       29.3   143
 4 Marinho       45.6   225
 5 Marinho       35.7   174
 6 Marinho       39.0   188
 7 Marinho       33.5   143
 8 Marinho       31.5   104
 9 A.doce        20.2   138
10 A.doce        39.7   365
# ... with 16 more rows

# Remover variáveis
dados %>% select(-Especie, - Ambiente)
# A tibble: 26 x 5
   ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
       <dbl> <int>       <dbl>   <dbl>       <dbl>
 1      38.9   320        2.08     0           6.7
 2      24.9   109        2.36    12.5        10.8
 3      29.3   143        2.31     4.5        10.3
 4      45.6   225        2.31     6          10.3
 5      35.7   174        2.31     6           8.9
 6      39.0   188        2.32     3.5         8.9
 7      33.5   143        2.37     0           9.4
 8      31.5   104        2.48     5           9.4
 9      20.2   138        2.16     1           9.2
10      39.7   365        2.04     4           9.2
# ... with 16 more rows

# Selecionar usando critérios nos nomes. Variáveis que começam com "A"...
dados %>% select(starts_with("A"))
# A tibble: 26 x 1
   Ambiente
   <fct>
 1 Marinho
```

```
 2 Marinho
 3 Marinho
 4 Marinho
 5 Marinho
 6 Marinho
 7 Marinho
 8 Marinho
 9 A.doce
10 A.doce
# ... with 16 more rows

# ... ou que conhenham "a" ...
dados %>% select(contains("a"))
# A tibble: 26 x 4
   Ambiente Massa Rotacao Temperatura
   <fct>    <int>   <dbl>       <dbl>
 1 Marinho    320     0           6.7
 2 Marinho    109    12.5        10.8
 3 Marinho    143     4.5        10.3
 4 Marinho    225     6          10.3
 5 Marinho    174     6           8.9
 6 Marinho    188     3.5         8.9
 7 Marinho    143     0           9.4
 8 Marinho    104     5           9.4
 9 A.doce     138     1           9.2
10 A.doce     365     4           9.2
# ... with 16 more rows

# ... ou que contenham o termo "O2" ...
dados %>% select(matches("O2"))
# A tibble: 26 x 2
   ConsumoO2 ConsumoO2kg
       <dbl>       <dbl>
 1      38.9        2.08
 2      24.9        2.36
 3      29.3        2.31
 4      45.6        2.31
 5      35.7        2.31
 6      39.0        2.32
 7      33.5        2.37
 8      31.5        2.48
 9      20.2        2.16
10      39.7        2.04
# ... with 16 more rows
```

4.4.3 rename() - Renomear colunas

```
# Renomear variável, na forma novo_name = name_antigo
dados %>% rename(Massa_g = Massa)
# A tibble: 26 x 7
   Especie      Ambiente ConsumoO2 Massa_g ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl>   <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       38.9     320        2.08     0           6.7
```

```
 2 C.sardinella Marinho       24.9     109        2.36    12.5        10.8
 3 C.sardinella Marinho       29.3     143        2.31     4.5        10.3
 4 C.sardinella Marinho       45.6     225        2.31     6          10.3
 5 C.sardinella Marinho       35.7     174        2.31     6           8.9
 6 C.sardinella Marinho       39.0     188        2.32     3.5         8.9
 7 C.sardinella Marinho       33.5     143        2.37     0           9.4
 8 C.sardinella Marinho       31.5     104        2.48     5           9.4
 9 C.sardinella A.doce        20.2     138        2.16     1           9.2
10 C.sardinella A.doce        39.7     365        2.04     4           9.2
# ... with 16 more rows

# Renomear todas as variáveis...
dados %>% rename_all(paste0, "_Coregonus")
# A tibble: 26 x 7
   Especie_Coregon~ Ambiente_Corego~ ConsumoO2_Coreg~ Massa_Coregonus ConsumoO2kg_Cor~
   <chr>            <fct>                      <dbl>           <int>            <dbl>
 1 C.sardinella     Marinho                     38.9             320             2.08
 2 C.sardinella     Marinho                     24.9             109             2.36
 3 C.sardinella     Marinho                     29.3             143             2.31
 4 C.sardinella     Marinho                     45.6             225             2.31
 5 C.sardinella     Marinho                     35.7             174             2.31
 6 C.sardinella     Marinho                     39.0             188             2.32
 7 C.sardinella     Marinho                     33.5             143             2.37
 8 C.sardinella     Marinho                     31.5             104             2.48
 9 C.sardinella     A.doce                      20.2             138             2.16
10 C.sardinella     A.doce                      39.7             365             2.04
# ... with 16 more rows, and 2 more variables: Rotacao_Coregonus <dbl>,
#   Temperatura_Coregonus <dbl>

# ... ou estão renomear variáveis de maneira condicional
dados %>% rename_if(is.factor, paste0, "_Factor")
# A tibble: 26 x 7
   Especie      Ambiente_Factor ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>                <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho               38.9   320        2.08     0           6.7
 2 C.sardinella Marinho               24.9   109        2.36    12.5        10.8
 3 C.sardinella Marinho               29.3   143        2.31     4.5        10.3
 4 C.sardinella Marinho               45.6   225        2.31     6          10.3
 5 C.sardinella Marinho               35.7   174        2.31     6           8.9
 6 C.sardinella Marinho               39.0   188        2.32     3.5         8.9
 7 C.sardinella Marinho               33.5   143        2.37     0           9.4
 8 C.sardinella Marinho               31.5   104        2.48     5           9.4
 9 C.sardinella A.doce                20.2   138        2.16     1           9.2
10 C.sardinella A.doce                39.7   365        2.04     4           9.2
# ... with 16 more rows
```

4.5 mutate() e transmute() - Criar novas colunas

As funções **mutate()** e **transmute()** permitem adicionar novas variáveis que são funções das variáveis existentes, ou seja, as funções aplicadas retornam vetores. A principal diferença entre elas é que a função **mutate()** mantém a tabela de entrada e gera novas colunas, enquanto que, a função **transmute()** gera novas colunas excluíndo o restante das colunas.

```r
# Funções principais
mutate(.data, ...)
transmute(.data, ...)

# Variações
mutate_all(.data, .funs, ...)
mutate_if(.data, .predicate, .funs, ...)
mutate_at(.data, .vars, .funs, ...)
transmute_all(.data, .funs, ...)
transmute_if(.data, .predicate, .funs, ...)
transmute_at(.data, .vars, .funs, ...)

# Principais funções associadas
as.numeric(), as.character(), as.factor()
abs(), sqrt(), log()
cos(x), sin(x), tan(x)
ifelse()
recode(), recode_factor()
```

4.5.1 mutate() - Criar novas colunas

```r
# Gerar nova coluna com valores iguais
dados %>% mutate(Nova = 12)
# A tibble: 26 x 8
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura  Nova
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl> <dbl>
 1 C.sardinella Marinho       38.9   320        2.08     0           6.7    12
 2 C.sardinella Marinho       24.9   109        2.36    12.5        10.8    12
 3 C.sardinella Marinho       29.3   143        2.31     4.5        10.3    12
 4 C.sardinella Marinho       45.6   225        2.31     6          10.3    12
 5 C.sardinella Marinho       35.7   174        2.31     6           8.9    12
 6 C.sardinella Marinho       39.0   188        2.32     3.5         8.9    12
 7 C.sardinella Marinho       33.5   143        2.37     0           9.4    12
 8 C.sardinella Marinho       31.5   104        2.48     5           9.4    12
 9 C.sardinella A.doce        20.2   138        2.16     1           9.2    12
10 C.sardinella A.doce        39.7   365        2.04     4           9.2    12
# ... with 16 more rows

# Gerar nova coluna com log da variável Massa
dados %>% mutate(MassaLog = log(Massa))
# A tibble: 26 x 8
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura MassaLog
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>    <dbl>
 1 C.sardinella Marinho       38.9   320        2.08     0           6.7     5.77
 2 C.sardinella Marinho       24.9   109        2.36    12.5        10.8     4.69
 3 C.sardinella Marinho       29.3   143        2.31     4.5        10.3     4.96
 4 C.sardinella Marinho       45.6   225        2.31     6          10.3     5.42
 5 C.sardinella Marinho       35.7   174        2.31     6           8.9     5.16
 6 C.sardinella Marinho       39.0   188        2.32     3.5         8.9     5.24
 7 C.sardinella Marinho       33.5   143        2.37     0           9.4     4.96
 8 C.sardinella Marinho       31.5   104        2.48     5           9.4     4.64
 9 C.sardinella A.doce        20.2   138        2.16     1           9.2     4.93
10 C.sardinella A.doce        39.7   365        2.04     4           9.2     5.90
```

```
# ... with 16 more rows

# Gerar nova coluna substituindo a variável original
dados %>% mutate(Ambiente = ifelse(Ambiente=="Marinho", yes = "M", no = "D"))
# A tibble: 26 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <chr>        <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella M             38.9   320        2.08     0           6.7
 2 C.sardinella M             24.9   109        2.36    12.5        10.8
 3 C.sardinella M             29.3   143        2.31     4.5        10.3
 4 C.sardinella M             45.6   225        2.31     6          10.3
 5 C.sardinella M             35.7   174        2.31     6           8.9
 6 C.sardinella M             39.0   188        2.32     3.5         8.9
 7 C.sardinella M             33.5   143        2.37     0           9.4
 8 C.sardinella M             31.5   104        2.48     5           9.4
 9 C.sardinella D             20.2   138        2.16     1           9.2
10 C.sardinella D             39.7   365        2.04     4           9.2
# ... with 16 more rows

# Transformar todas variáveis numéricas
dados %>% mutate_if(is.numeric, log)
# A tibble: 26 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <dbl>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       3.66  5.77       0.734 -Inf          1.90
 2 C.sardinella Marinho       3.21  4.69       0.858    2.53       2.38
 3 C.sardinella Marinho       3.38  4.96       0.838    1.50       2.33
 4 C.sardinella Marinho       3.82  5.42       0.836    1.79       2.33
 5 C.sardinella Marinho       3.57  5.16       0.838    1.79       2.19
 6 C.sardinella Marinho       3.66  5.24       0.840    1.25       2.19
 7 C.sardinella Marinho       3.51  4.96       0.863 -Inf          2.24
 8 C.sardinella Marinho       3.45  4.64       0.909    1.61       2.24
 9 C.sardinella A.doce        3.01  4.93       0.772    0          2.22
10 C.sardinella A.doce        3.68  5.90       0.711    1.39       2.22
# ... with 16 more rows

# Especificar qual variável transformar...
dados %>% mutate_at(vars(Massa, ConsumoO2), log)
# A tibble: 26 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <dbl>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       3.66  5.77        2.08     0           6.7
 2 C.sardinella Marinho       3.21  4.69        2.36    12.5        10.8
 3 C.sardinella Marinho       3.38  4.96        2.31     4.5        10.3
 4 C.sardinella Marinho       3.82  5.42        2.31     6          10.3
 5 C.sardinella Marinho       3.57  5.16        2.31     6           8.9
 6 C.sardinella Marinho       3.66  5.24        2.32     3.5         8.9
 7 C.sardinella Marinho       3.51  4.96        2.37     0           9.4
 8 C.sardinella Marinho       3.45  4.64        2.48     5           9.4
 9 C.sardinella A.doce        3.01  4.93        2.16     1           9.2
10 C.sardinella A.doce        3.68  5.90        2.04     4           9.2
# ... with 16 more rows
```

```
# ... ou buscar quais variáveis transformar
dados %>% mutate_at(vars(contains("O2")), log)
# A tibble: 26 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       3.66   320       0.734     0           6.7
 2 C.sardinella Marinho       3.21   109       0.858    12.5        10.8
 3 C.sardinella Marinho       3.38   143       0.838     4.5        10.3
 4 C.sardinella Marinho       3.82   225       0.836     6          10.3
 5 C.sardinella Marinho       3.57   174       0.838     6           8.9
 6 C.sardinella Marinho       3.66   188       0.840     3.5         8.9
 7 C.sardinella Marinho       3.51   143       0.863     0           9.4
 8 C.sardinella Marinho       3.45   104       0.909     5           9.4
 9 C.sardinella A.doce        3.01   138       0.772     1           9.2
10 C.sardinella A.doce        3.68   365       0.711     4           9.2
# ... with 16 more rows
```

4.5.2 transmute() - Criar novas colunas excluíndo o restante

```
# Retornar apenas as variáveis Especie e log da Massa
dados %>% transmute(Especie, LogMassa = log(Massa))
# A tibble: 26 x 2
   Especie      LogMassa
   <chr>           <dbl>
 1 C.sardinella     5.77
 2 C.sardinella     4.69
 3 C.sardinella     4.96
 4 C.sardinella     5.42
 5 C.sardinella     5.16
 6 C.sardinella     5.24
 7 C.sardinella     4.96
 8 C.sardinella     4.64
 9 C.sardinella     4.93
10 C.sardinella     5.90
# ... with 16 more rows
```

4.6 arrange() - Altera a ordem das linhas

As função **arrange()** e a função auxiliar **desc()** são usadas para ordenar as tabelas de dados em função de uma ou mais variáveis.

```
# Funções principais
arrange(.data, ...)
desc()
```

```
# Ordenar de maneira crescente ...
dados %>% arrange(Temperatura)
# A tibble: 26 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       38.9   320        2.08     0           6.7
 2 C.autumnalis Marinho       24.6   130        2.28     0           6.7
 3 C.autumnalis Marinho       36.9   157        2.37     0           7.2
```

```
 4 C.autumnalis Marinho       13.5   124        2.04     0          7.2
 5 C.autumnalis Marinho       32.3   234        2.14     5          7.2
 6 C.sardinella A.doce        41.6   362        2.06     5          8.3
 7 C.sardinella Marinho       35.7   174        2.31     6          8.9
 8 C.sardinella Marinho       39.0   188        2.32     3.5        8.9
 9 C.sardinella A.doce        27.9   231        2.08     3          8.9
10 C.sardinella A.doce        30.5   216        2.15     2.5        8.9
# ... with 16 more rows

# ... ou ordenar de maneira descrecente
dados %>% arrange(desc(ConsumoO2))
# A tibble: 26 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       45.6   225        2.31     6          10.3
 2 C.sardinella A.doce        41.6   362        2.06     5          8.3
 3 C.sardinella A.doce        39.7   365        2.04     4          9.2
 4 C.sardinella Marinho       39.0   188        2.32     3.5        8.9
 5 C.sardinella Marinho       38.9   320        2.08     0          6.7
 6 C.sardinella A.doce        37.4   183        2.31    10          10.6
 7 C.autumnalis Marinho       36.9   157        2.37     0          7.2
 8 C.sardinella A.doce        36.7   256        2.16     5.5        9.7
 9 C.sardinella Marinho       35.7   174        2.31     6          8.9
10 C.sardinella Marinho       33.5   143        2.37     0          9.4
# ... with 16 more rows

# Ordenar combinando mais de um critério
dados %>% arrange(Ambiente, desc(Temperatura))
# A tibble: 26 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella A.doce        25.8   140        2.27     8          10.8
 2 C.sardinella A.doce        37.4   183        2.31    10          10.6
 3 C.sardinella A.doce        36.7   256        2.16     5.5        9.7
 4 C.sardinella A.doce        11.0   135        1.91     2          9.7
 5 C.sardinella A.doce        33.4   221        2.18     5          9.4
 6 C.sardinella A.doce        29.1   204        2.15     7          9.4
 7 C.sardinella A.doce        20.3   180        2.05     5          9.4
 8 C.sardinella A.doce        20.2   138        2.16     1          9.2
 9 C.sardinella A.doce        39.7   365        2.04     4          9.2
10 C.sardinella A.doce        16.8   129        2.11     7.5        9.2
# ... with 16 more rows
```

4.7 group_by() - Agrupar linhas

A função **group_by()** permite agrupar linhas/obervações/registros das tabelas em grupos definidos por outras variáveis. A função retorna objeto que pertencem simultaneamente as classes *"grouped_df"*, *"tbl_df"* e *"data.frame"*, e estes, podem ser utilizados para aplicar funções por grupos. A função **ungroup()** permite remover um agrupamento prévio.

```
# Funções principais
group_by(.data, ...)
ungroup(x, ...)
```

```r
# Variações
group_by_all(.data, .funs = list(), ...)
group_by_if(.data, .vars, .funs = list(), ...)
group_by_at(.data, .predicate, .funs = list(), ...)
```

```r
# Agrupar observações pela variável Especie
dados %>% group_by(Especie)
# A tibble: 26 x 7
# Groups:   Especie [2]
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       38.9   320        2.08     0           6.7
 2 C.sardinella Marinho       24.9   109        2.36    12.5        10.8
 3 C.sardinella Marinho       29.3   143        2.31     4.5        10.3
 4 C.sardinella Marinho       45.6   225        2.31     6          10.3
 5 C.sardinella Marinho       35.7   174        2.31     6           8.9
 6 C.sardinella Marinho       39.0   188        2.32     3.5         8.9
 7 C.sardinella Marinho       33.5   143        2.37     0           9.4
 8 C.sardinella Marinho       31.5   104        2.48     5           9.4
 9 C.sardinella A.doce        20.2   138        2.16     1           9.2
10 C.sardinella A.doce        39.7   365        2.04     4           9.2
# ... with 16 more rows

# Agrupar observações com mais de uma variável
dados %>% group_by(Ambiente, Especie)
# A tibble: 26 x 7
# Groups:   Ambiente, Especie [3]
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       38.9   320        2.08     0           6.7
 2 C.sardinella Marinho       24.9   109        2.36    12.5        10.8
 3 C.sardinella Marinho       29.3   143        2.31     4.5        10.3
 4 C.sardinella Marinho       45.6   225        2.31     6          10.3
 5 C.sardinella Marinho       35.7   174        2.31     6           8.9
 6 C.sardinella Marinho       39.0   188        2.32     3.5         8.9
 7 C.sardinella Marinho       33.5   143        2.37     0           9.4
 8 C.sardinella Marinho       31.5   104        2.48     5           9.4
 9 C.sardinella A.doce        20.2   138        2.16     1           9.2
10 C.sardinella A.doce        39.7   365        2.04     4           9.2
# ... with 16 more rows

# Desagrupar
dados %>% group_by(Especie) %>% ungroup()
# A tibble: 26 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       38.9   320        2.08     0           6.7
 2 C.sardinella Marinho       24.9   109        2.36    12.5        10.8
 3 C.sardinella Marinho       29.3   143        2.31     4.5        10.3
 4 C.sardinella Marinho       45.6   225        2.31     6          10.3
 5 C.sardinella Marinho       35.7   174        2.31     6           8.9
 6 C.sardinella Marinho       39.0   188        2.32     3.5         8.9
 7 C.sardinella Marinho       33.5   143        2.37     0           9.4
```

```
 8 C.sardinella Marinho       31.5   104        2.48     5           9.4
 9 C.sardinella A.doce        20.2   138        2.16     1           9.2
10 C.sardinella A.doce        39.7   365        2.04     4           9.2
# ... with 16 more rows
```

4.8 summarize() - Obter estatísticas descritivas

A função **summarise()** permite calcular estatísticas nas variáveis, reduzindo assim vários valores de um vetor em um único valor de estátistica descritiva.

```
# Função principais
summarise(.data, ...)

# Variações
summarise_all(.data, .funs, ...)
summarise_if(.data, .predicate, .funs, ...)
summarise_at(.data, .vars, .funs, ...)

# Principais funções associadas
mean(), median()
sd(), IQR()
min(), max(), quantile()
first(), last(), nth()
n(), n_distinct()
any(), all()
```

```
# Calcular estatística média para uma única variável
dados %>% summarise(mean(Temperatura))
# A tibble: 1 x 1
  `mean(Temperatura)`
                <dbl>
1                9.03

# Calcular simultaneamente vários tipo de estatística
dados %>% summarise(mean(Temperatura), sd(Massa))
# A tibble: 1 x 2
  `mean(Temperatura)` `sd(Massa)`
                <dbl>       <dbl>
1                9.03        72.6

# Calcular condicionando ao tipo de variável
dados %>% summarise_if(is.numeric, mean)
# A tibble: 1 x 5
  ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
      <dbl> <dbl>       <dbl>   <dbl>       <dbl>
1      30.1   190        2.20    4.42        9.03

# A função só permite retornar um único valore por coluna, então...
dados %>% summarise_if(is.numeric, quantile, prob = c(0.25))
# A tibble: 1 x 5
  ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
      <dbl> <dbl>       <dbl>   <dbl>       <dbl>
1      25.0  136.        2.09    2.12         8.9
```

4.9 Agrupar linha e colunas

O pacote *dplyr* também reimplementa as funções básicas para agrupar linhas e colunas aos *data.frame*. A função **add_row** adiciona linhas usando o nome das variáveis e as função **bind_rows** e **bind_cols** agrupam linhas e colunas respectivamente.

```
# Funções principais
add_row(.data, ...)
bind_rows(...)
bind_cols(...)
```

```
# Adicionar registro usando nomes de algumas variáveis
# Note que as variáveis não especificadas são atributidas como NAs
dados %>% add_row(Especie = "C.autumnalis", Ambiente = "Marinho", Massa = 200)
# A tibble: 27 x 7
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura
   <chr>        <chr>        <dbl> <dbl>       <dbl>   <dbl>       <dbl>
 1 C.sardinella Marinho       38.9   320        2.08     0           6.7
 2 C.sardinella Marinho       24.9   109        2.36    12.5        10.8
 3 C.sardinella Marinho       29.3   143        2.31     4.5        10.3
 4 C.sardinella Marinho       45.6   225        2.31     6          10.3
 5 C.sardinella Marinho       35.7   174        2.31     6           8.9
 6 C.sardinella Marinho       39.0   188        2.32     3.5         8.9
 7 C.sardinella Marinho       33.5   143        2.37     0           9.4
 8 C.sardinella Marinho       31.5   104        2.48     5           9.4
 9 C.sardinella A.doce        20.2   138        2.16     1           9.2
10 C.sardinella A.doce        39.7   365        2.04     4           9.2
# ... with 17 more rows

# Dados de exemplo para adicionar a tabela
Sequencia <- data.frame(Seq = seq_len(nrow(dados)))
# Note que esse comando abaixo já atribui o resultado ao objeto original
Sequencia %<>% as.tbl()
Sequencia
# A tibble: 26 x 1
     Seq
   <int>
 1     1
 2     2
 3     3
 4     4
 5     5
 6     6
 7     7
 8     8
 9     9
10    10
# ... with 16 more rows

# Agrupar colunas
dados %>% bind_cols(Sequencia)
# A tibble: 26 x 8
   Especie      Ambiente ConsumoO2 Massa ConsumoO2kg Rotacao Temperatura   Seq
   <chr>        <fct>        <dbl> <int>       <dbl>   <dbl>       <dbl> <int>
```

```
 1 C.sardinella Marinho       38.9   320        2.08     0          6.7     1
 2 C.sardinella Marinho       24.9   109        2.36    12.5       10.8     2
 3 C.sardinella Marinho       29.3   143        2.31     4.5       10.3     3
 4 C.sardinella Marinho       45.6   225        2.31     6         10.3     4
 5 C.sardinella Marinho       35.7   174        2.31     6          8.9     5
 6 C.sardinella Marinho       39.0   188        2.32     3.5        8.9     6
 7 C.sardinella Marinho       33.5   143        2.37     0          9.4     7
 8 C.sardinella Marinho       31.5   104        2.48     5          9.4     8
 9 C.sardinella A.doce        20.2   138        2.16     1          9.2     9
10 C.sardinella A.doce        39.7   365        2.04     4          9.2    10
# ... with 16 more rows
```

4.10 Combinar várias funções e salvar objetos

No processamento de dados é comum combinar várias funções em sequência. Usando as funções do pacote *dplyr* junto com o operador *pipe* é possível combinar as etapas do processamento de dados de uma maneira simplificada. Por exemplo, usando as funções **filter** e **select** é possível selecionar linhas e colunas em um única etapa, as funções **group_by** e **summarise** podem ser combinadas para calcular estatísticas para grupos, e ainda usando as funções **group_by** e **mutate** é possivel gerar novas variáveis calculadas por grupos. Alguns exemplos:

```
# Selecionar apenas as variáveis Especie, Massa e ConsumoO2, aplicar log ...
# ... e então renomear variáveis numéricas
dados %>%
    select(Especie, Massa, ConsumoO2) %>%
    mutate_if(is.numeric, log, base = 2) %>%
    rename_if(is.numeric, paste0, "_log")
# A tibble: 26 x 3
   Especie      Massa_log ConsumoO2_log
   <chr>            <dbl>         <dbl>
 1 C.sardinella      8.32          5.28
 2 C.sardinella      6.77          4.64
 3 C.sardinella      7.16          4.87
 4 C.sardinella      7.81          5.51
 5 C.sardinella      7.44          5.16
 6 C.sardinella      7.55          5.28
 7 C.sardinella      7.16          5.07
 8 C.sardinella      6.70          4.98
 9 C.sardinella      7.11          4.34
10 C.sardinella      8.51          5.31
# ... with 16 more rows

# Agrupar por Especie, e então, calcular média e número de obervações por grupo
dados %>%
    group_by(Especie) %>%
    summarise(mean(Rotacao), n())
# A tibble: 2 x 3
  Especie      `mean(Rotacao)` `n()`
  <chr>                  <dbl> <int>
1 C.autumnalis            1.25     4
2 C.sardinella            5       22

# Remover algumas variáveis, agrupar por Ambiente...
# ... e então calcular a rotação média por ambiente
```

```r
# Note que a variável RotacaoMediaPorAmbiente é um vetor calculado por ambiente
dados %>%
    select(-ConsumoO2, -ConsumoO2kg, -Massa) %>%
    group_by(Ambiente) %>%
    mutate(RotacaoMediaPorAmbiente = mean(Rotacao))
# A tibble: 26 x 5
# Groups:   Ambiente [2]
   Especie      Ambiente Rotacao Temperatura RotacaoMediaPorAmbiente
   <chr>        <fct>      <dbl>       <dbl>                   <dbl>
 1 C.sardinella Marinho      0           6.7                    3.54
 2 C.sardinella Marinho     12.5        10.8                    3.54
 3 C.sardinella Marinho      4.5        10.3                    3.54
 4 C.sardinella Marinho      6          10.3                    3.54
 5 C.sardinella Marinho      6           8.9                    3.54
 6 C.sardinella Marinho      3.5         8.9                    3.54
 7 C.sardinella Marinho      0           9.4                    3.54
 8 C.sardinella Marinho      5           9.4                    3.54
 9 C.sardinella A.doce       1           9.2                    5.18
10 C.sardinella A.doce       4           9.2                    5.18
# ... with 16 more rows

# Filtrar apenas espécie C.sardinella, remover algumas variáveis, agrupar por Ambiente...
# ... e então calcular temperatura média por ambiente e desvio da temperatura média
dados %>%
    filter(Especie == "C.sardinella") %>%
    select(-ConsumoO2kg, -Rotacao, - Massa) %>%
    group_by(Ambiente) %>%
    mutate(TemperaturaMedia = mean(Temperatura)) %>%
    mutate(DesvioTemperatura = Temperatura-TemperaturaMedia)
# A tibble: 22 x 6
# Groups:   Ambiente [2]
   Especie      Ambiente ConsumoO2 Temperatura TemperaturaMedia DesvioTemperatura
   <chr>        <fct>        <dbl>       <dbl>            <dbl>             <dbl>
 1 C.sardinella Marinho       38.9         6.7             9.34            -2.64
 2 C.sardinella Marinho       24.9        10.8             9.34             1.46
 3 C.sardinella Marinho       29.3        10.3             9.34             0.963
 4 C.sardinella Marinho       45.6        10.3             9.34             0.963
 5 C.sardinella Marinho       35.7         8.9             9.34            -0.438
 6 C.sardinella Marinho       39.0         8.9             9.34            -0.438
 7 C.sardinella Marinho       33.5         9.4             9.34             0.0625
 8 C.sardinella Marinho       31.5         9.4             9.34             0.0625
 9 C.sardinella A.doce        20.2         9.2             9.42            -0.221
10 C.sardinella A.doce        39.7         9.2             9.42            -0.221
# ... with 12 more rows

# Note que em todos os exemplos desse tutorial o objeto dados são foi alterado...
# ... então ainda é preciso salvar os objetos depois do precessamento
dados_processados <- dados %>%
    filter(Especie == "C.sardinella") %>%
    select(-ConsumoO2kg, -Rotacao, - Massa) %>%
    group_by(Ambiente) %>%
    mutate(TemperaturaMedia = mean(Temperatura)) %>%
    mutate(DesvioTemperatura = Temperatura-TemperaturaMedia) %>%
```

```
    as.data.frame()

# Visualizar o resultado do processamento
dados_processados
```

	Especie	Ambiente	ConsumoO2	Temperatura	TemperaturaMedia	DesvioTemperatura
1	C.sardinella	Marinho	38.868	6.7	9.337500	-2.63750000
2	C.sardinella	Marinho	24.882	10.8	9.337500	1.46250000
3	C.sardinella	Marinho	29.292	10.3	9.337500	0.96250000
4	C.sardinella	Marinho	45.618	10.3	9.337500	0.96250000
5	C.sardinella	Marinho	35.694	8.9	9.337500	-0.43750000
6	C.sardinella	Marinho	38.976	8.9	9.337500	-0.43750000
7	C.sardinella	Marinho	33.498	9.4	9.337500	0.06250000
8	C.sardinella	Marinho	31.524	9.4	9.337500	0.06250000
9	C.sardinella	A.doce	20.196	9.2	9.421429	-0.22142857
10	C.sardinella	A.doce	39.690	9.2	9.421429	-0.22142857
11	C.sardinella	A.doce	25.812	10.8	9.421429	1.37857143
12	C.sardinella	A.doce	37.404	10.6	9.421429	1.17857143
13	C.sardinella	A.doce	33.396	9.4	9.421429	-0.02142857
14	C.sardinella	A.doce	29.100	9.4	9.421429	-0.02142857
15	C.sardinella	A.doce	36.654	9.7	9.421429	0.27857143
16	C.sardinella	A.doce	11.016	9.7	9.421429	0.27857143
17	C.sardinella	A.doce	20.340	9.4	9.421429	-0.02142857
18	C.sardinella	A.doce	41.616	8.3	9.421429	-1.12142857
19	C.sardinella	A.doce	27.858	8.9	9.421429	-0.52142857
20	C.sardinella	A.doce	30.546	8.9	9.421429	-0.52142857
21	C.sardinella	A.doce	16.758	9.2	9.421429	-0.22142857
22	C.sardinella	A.doce	25.320	9.2	9.421429	-0.22142857

5 Guia de ajuda rápida

```
# Operadores
%>% - Encadeamento (pipe)
%$% - Selecionar variáveis pelo nome
%<>% - Atribur diretamente
%T>% - Abrir braço no fluxo de encadeamento

# Tabelas tibble
as.tbl - Converter dada.frame em tbl_df
as.data.frame - Converte para data.frame

# Selecionar variáveis
vars - Selecionar/excluir variáveis pelo mome ou posição
starts_with - Selecionar variáveis cujo nome começam com um termo
ends_with - Selecionar variáveis cujo nome terminam com um termo
contains - Selecionar variáveis cujo nome contenham um termo
matches - Selecionar variáveis correspondem com uma expressão
num_range - Selecionar variáveis que correspondam a um intervalo numérico

# Funções principais
slice - Filtrar linhas usando posição
filter - Filtrar linhas com base em seus valores
select - Selecionar variáveis com base em seus nomes
pull - Selecionar variável e retornar vetor
```

```
rename - Remomear variáveis
mutate - Adicionar novas variáveis que são funções de variáveis existentes
transmute - Gerar novas variáveis excluíndo o restante das variáveis
arrange - Alterar a ordem das linhas
desc - Alterar a ordem das linhas de maneira decrescente
group_by - Agrupar linhas para aplicar funções por grupos
ungroup - Remover um agrupamento prévio
summarise - Reduzir vários valores para em estátisticas descritivas
add_row - Adicionar linhas usando o nome das variáveis
bind_rows - Agrupar linhas
bind_cols - Agrupar colunas

# Variações das funções
_all - Aplicar ou forçar a aplicação em todas as variáveis da tabela
_if - Aplicar as funções condicionada as variáveis
_at - Aplicar as funções a variáveis selecionadas

# Funções auxiliares
n - Contar número de observações
n_distinct - Contar número de observações únicas
recode - Recodificar character
recode_factor - Recodificar fatores
sample_n - Extrair n linhas aleatoriamete
sample_frac - Extrair proporção de linhas aleatoriamete
top_n - Extrair n linhas do maiores valores de determinada variável
between - Teste lógico. Valores pertencente a intervalo
near - Teste lógico. Comparar valores com tolerância
all_vars - Teste lógico em todas as variáveis. Aplicando intersecção dos resultados
any_vars - Teste lógico em todas as variáveis. Aplicando união do resultados
```

6 Conclusão

O objetivo deste texto foi apenas apresentar as funções básicas usadas no processamento de dados usando os pacotes *magrittr* e *dplyr*. Espero que este texto tenha sido útil e, por favor, avise-me se tiver dúvidas ou sugestões sobre este texto.

7 Mais informações

Outros textos e tutoriais sobre R podem ser encontrados em https://vanderleidebastiani.github.io/tutoriais.

8 Referências

- Bache, Stefan M.; Wickham, Hadley; 2014. **magrittr: A Forward-Pipe Operator for R**.
- Müller, Kirill; Wickham, Hadley; 2020. **tibble: Simple Data Frames**.
- R Core Team; 2018. **R Language Definition**. https://cran.r-project.org/doc/manuals/R-lang.html
- Tidyverse; 2020. **R packages for data science**. www.tidyverse.org
- Wickham, Hadley; François, Romain; Henry, Lionel; Müller, Kirill; 2020. **dplyr: A Grammar of Data Manipulation**.

- Wohlshag, Donald E.; 1957. **Differences in Metabolic Rates of Migratory and Resident Freshwater Forms of an Arctic Whitefish**. Ecology, Vol. 38, pp.502-510